University Of Bahrain
IT College, Computer Science Dept.
ITCS 215 Data Structures in C++
Quiz#1 Chapter 2: Inheritance & Composition

ID :
Name:
Section:

10

③
```
class Win7
{
private:
       string        Company;
       float         Version;
public:
       Win7( ) {
              Company="Microsoft";
              Version=7.0;
              cout<<Company;
       }

       Win7 (string   Org, float Ver ) {
              Origin  = Org;
              Version = Ver;
       }

       void print ( ) {
              cout<<"PO. "<< Company <<
              cout<<Version << endl;
       }
};


class  Win8: public  Win7
{
protected:
       bool          Touch;
       int           size;
public:

    ⑥ Win8 (string ,float , bool , int );
    ② void print();

};
```

Implement the following functions:

1) The Win8 *constructor* function with 4 parameters.

```
win8 :: win8 (string c, float v, bool t, int s),
                  : win7 (c, v)
{
   Touch = t;
   Size = s;
}
```

2) The print function in class Win8 that prints all 4 attributes

```
void win8 :: print ( )
{
  win7:: print ();
  cout << touch << size;
}
```

3) The output of the following

```
void  main ( )
{
   Win7        HomeEdition;


}
```

Microsoft

10

University Of Bahrain IT College, Computer Science Dept. ITCS 215 Data Structure Quiz#2 (Chapter 3)	ID : Name: Section:

Write a function (not a member function) called **operateOnLists** that accepts two arrayListType objects called ***L1 and L2*** as parameters and performs the following:

- The function will find the sum of all elements of object `L1`.
- Afterwards, the function inserts in `L2` the sum of `L1` at the end of `L2` if the sum is not found in `L2`. If the sum is found then remove it from `L2`.

(You may use functions `retrieveAt, removeAt, insertAt, listsize, insertEnd, isEmpty, isFull, remove, seqSearch, operator=`).

```
template<class Type>
void operateOnLists ( arrayListType<Type>& L1,
                      arrayListType<Type>&  L2 ); {
```

```
Type sum=0; Type n; int found;
for (int i=0 ; i<L1.listsize() ; i++){
   L1.retrieve(i,n);
   sum+=n;
}
found = L2.seqSearch(sum);
if (found != -1){
   L2.removeAt(found);}
else {L2.insertEnd(sum);}
}
```

University Of Bahrain IT College, Computer Science Dept. ITCS 215 Data Structure Quiz#4 (Chapter 7)	ID: Name: Section:

10

Write a function `upperBoundStack` that accepts a stack ***S*** of type `stackType` and `upperbound` of type `Type` as parameter. The function removes from the stack ***S*** all elements larger than `upperbound`. Your function must keep remaining elements in the stack in their original relative order.

Example:
Upperbound = 9

Stack **S** before function call : 5 10 17 3 8 15 2

Stack **S** after function call : 5 3 8 2

The function prototype is:

```
void upperBoundStack( StackType<Type>& S,Type upperBound);
```

```
{ stackType<Type> tmp;

  while (!S.isEmpty()){
    if (S.top() >upperBound)
        { S.pop();}
    else { tmp.push(S.top());
           S.pop();}
  }
  while (!tmp.isEmpty())
  {
    S.push(tmp.top());
    tmp.pop();
  }
}
```